22 | 掌控数据：家里的数据可以怎么利用?

2021-01-01 郭朝斌

物联网开发实战 进入课程 >

讲述：郭朝斌

时长 20:49 大小 19.07M

你好，我是郭朝斌。

到目前为止，我们已经完成智能电灯、光照传感器、智能音箱和自动浇花器的实战训练，在这个过程中，我们主要关注的是设备功能和远程控制的实现。

其实，物联网设备会生成大量的数据。如果我们能把这些数据存储到物联网系统的数据库中，并且好好应用这些数据，比如提供查询和分析功能，就能够产出更大的价值。

这一讲，我就基于自动浇花器来讲一讲数据的应用方法，主要包括以下两种：

1. 基于腾讯云物联网平台提供的**数据流**功能，介绍一个**设备消息推送应用**的配置方法。

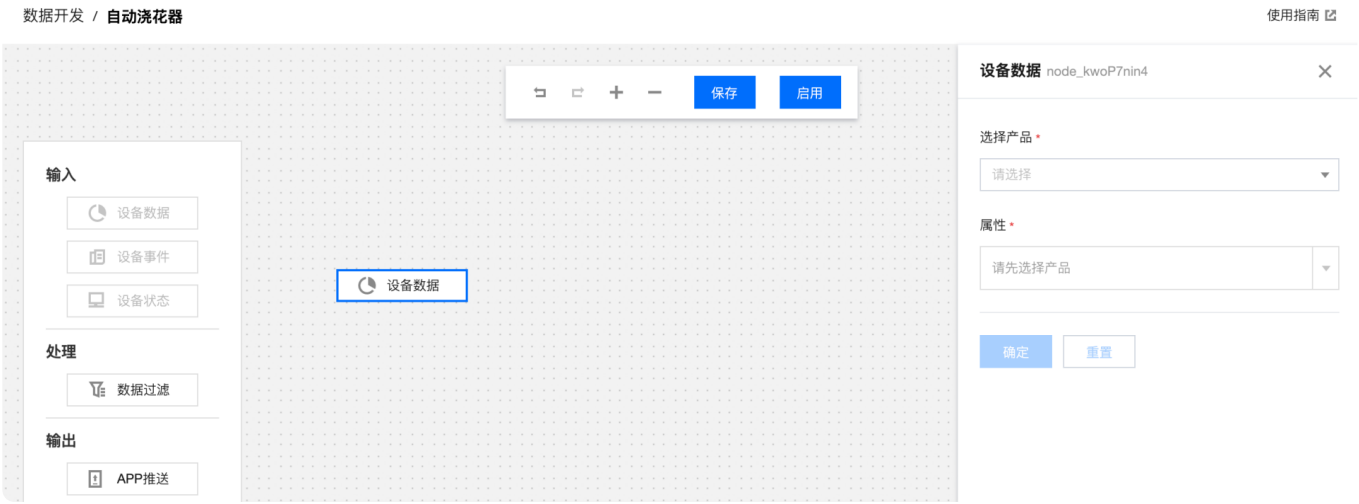
2. 基于腾讯云的 **HTTP 方式的数据同步**功能，开发一个 **Web 数据应用系统**。因为需要购买云服务器，所以你可以酌情选择是否实际部署。

方法一：基于数据流的设备消息推送应用

腾讯云物联网平台已经为我们提供了一种简便的数据应用方法。我们可以使用它的可视化编辑界面来完成数据流的创建工作。

你可以登录腾讯云物联网平台的控制台，然后进入我们之前创建的“智能家居”项目，点击左边菜单栏中的“数据开发”。

然后，你需要新建一个数据流，名称可以是“自动浇花器”。点击数据流列表中的“自动浇花器”项目，你就可以进入可视化的编辑界面。

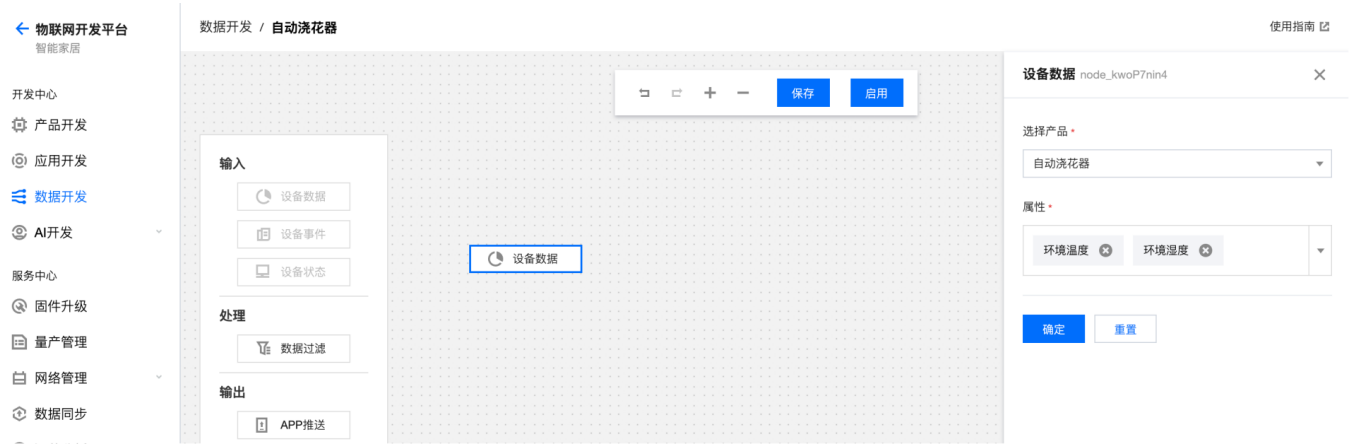


在可视化编辑界面，我们可以看到，一个数据流包括“输入”“处理”和“输出”三个部分。

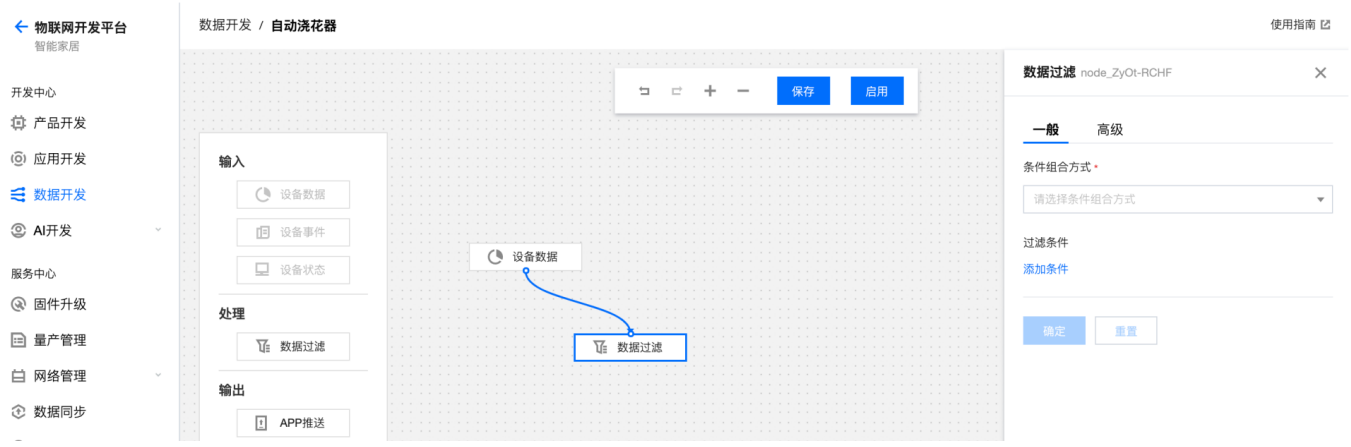
1. 输入，包括设备数据、设备事件和设备状态三种，其中设备数据和设备事件与物模型中的定义是一致的。设备状态是设备的上线、下线的状态变化。
2. 处理，可以编写基本的判断逻辑来过滤输入数据。
3. 输出，可以作为消息将数据推送到 App 或者小程序中。你可以对消息的内容模板进行定义。

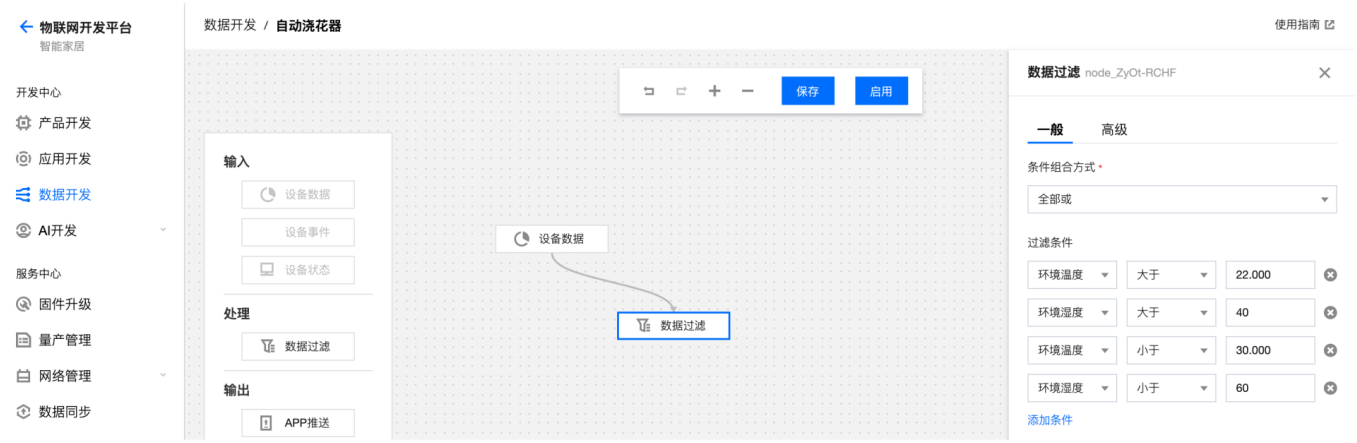
自动浇花器设备会上报环境的温度、湿度信息，那么我们可以定义一个温湿度不适宜的消息提醒。

你可以拖拽“设备数据”到编辑区域，然后点击这个模块，在右边的选项中定义设备数据，产品选择“自动浇花器”，属性选择“环境温度”和“环境湿度”。



接着，你可以添加“数据过滤”模块，并且将这个模块与“设备数据”模块相连，然后点击这个模块，在右边编辑过滤条件。

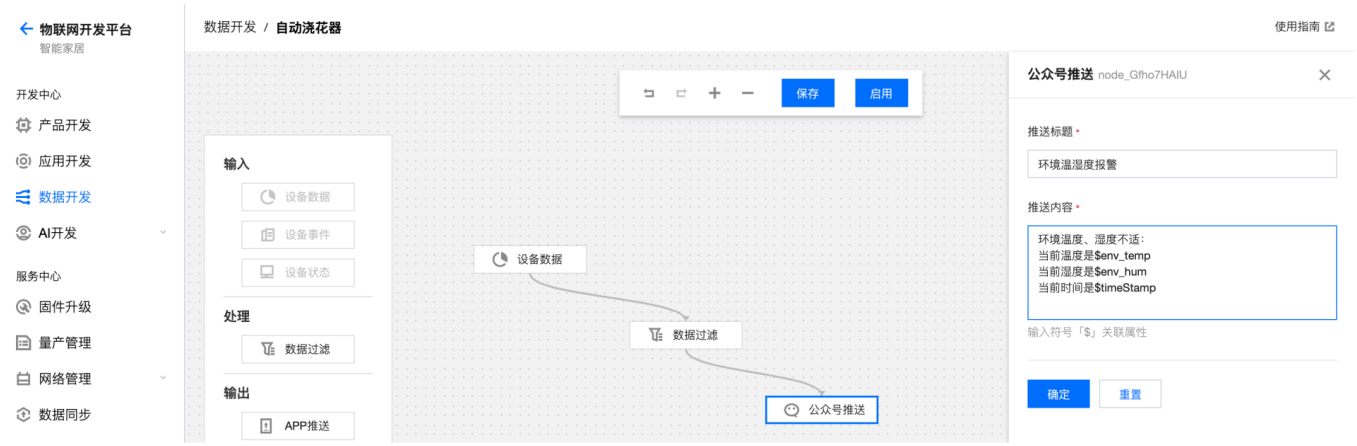




完成数据过滤的定义后，你需要继续添加“公众号推送”模块，并且编辑消息推送的模板。具体的消息模板定义，你可以参考下面内容：

复制代码

```
环境温度、湿度不适宜:
当前温度是$env_temp
当前湿度是$env_hum
当前时间是$timeStamp
```



最后，点击页面上方的“保存”和“启用”，你就完成了数据流的定义。

如果你的自动浇花器设备是在线状态，那么当环境的温度或者湿度过高、过低时，你就会在腾讯连连小程序的消息列表中收到“告警”消息。

方法二：基于 HTTP 数据同步的 Web 数据应用

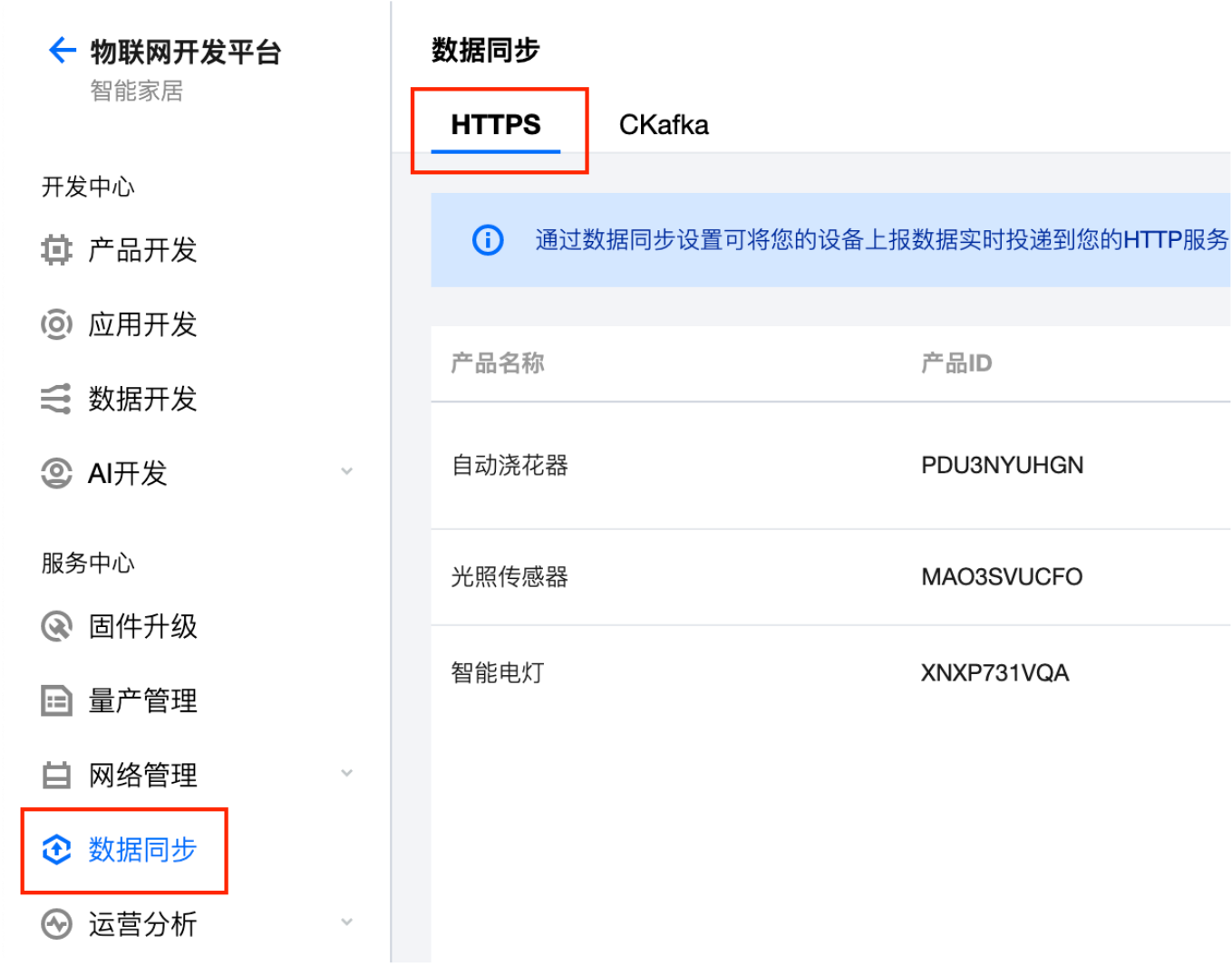
物联网平台除了提供数据流的方式，还可以基于 HTTP 协议把数据推送到你指定的网址，比如你开发的 Web 服务器的网址。所以你也可以使用这种方式，更加灵活地利用物联网设备的数据。

数据同步体验

我们借助在线的 Webhook 服务，测试一下数据同步功能。

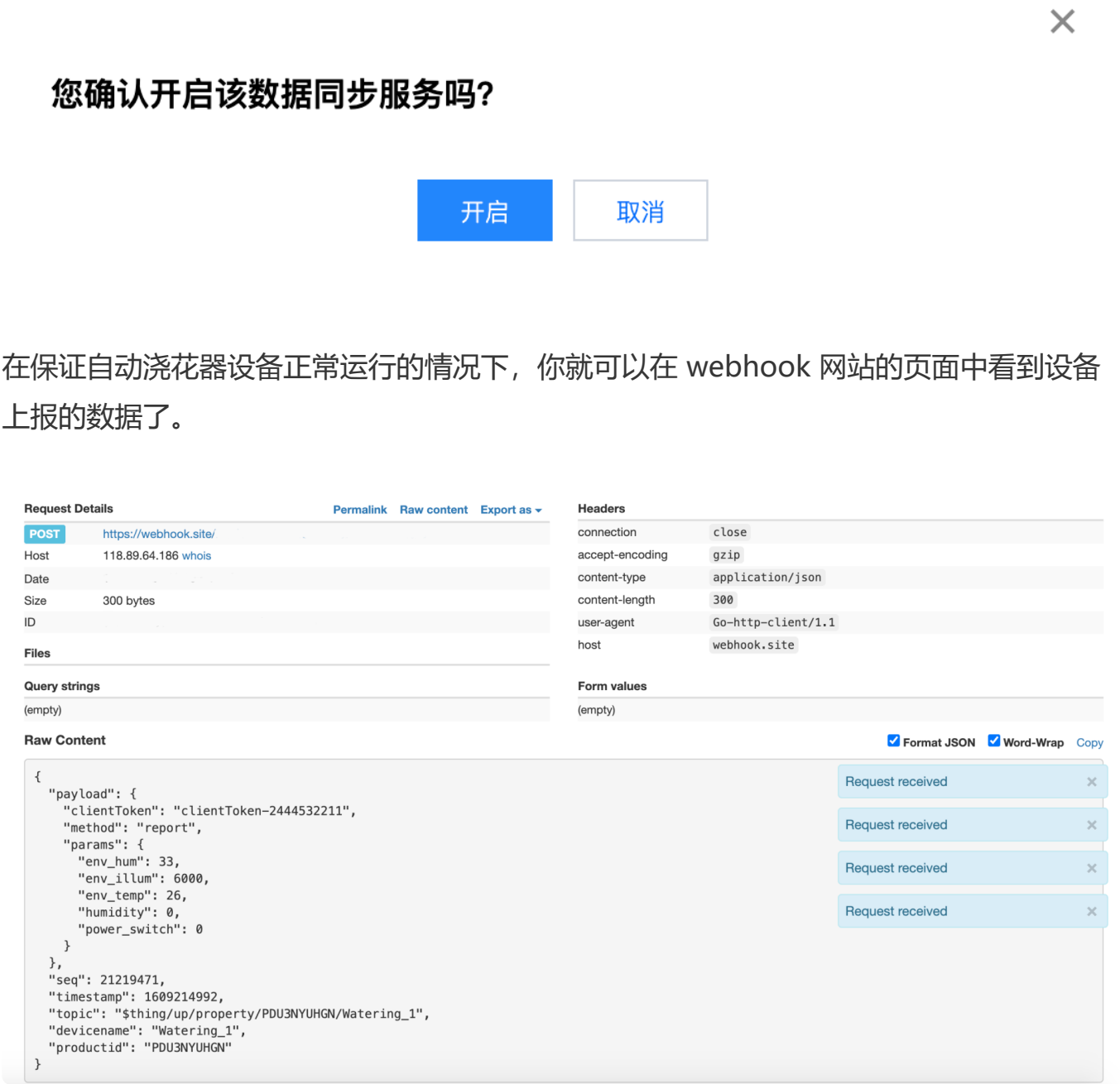
首先，打开webhook.site网站，并且记录页面中显示的专属 URL 地址。

然后，你需要登录腾讯云物联网平台，进入我们之前创建的“智能家居”项目，再点击左边菜单栏中的“数据同步”，选择“HTTPS”标签。



接着，点击“自动浇花器”对应的“设置”链接，在设置窗口，将 webhook 网站获取到的 URL 地址，粘贴在输入框。

完成设置后，打开“生效状态”。



下面我来介绍一下用 Python 语言来开发 Web 服务器的方法。

因为购买云服务器需要费用，所以这个实战任务是选学内容，你可以在自己的电脑上实践一下 Web 服务器的开发过程，然后酌情考虑要不要部署到云服务器上。

准备 Django 开发环境

Django 是一个流行的基于 Python 语言的 Web 开发框架。它提供了强大的功能，同时也简单易用。接下来，我们就基于 Django 来实现一个 Web 应用程序。

首先是在电脑上配置 Django 的开发环境，在终端上运行下面的命令安装 Django，准备好需要用到的工具。

```
$ pip3 install django
```

复制代码

安装完成后，你可以在终端上切换到一个代码开发目录，比如：

```
$ cd ~/study/iot/geektime
```

复制代码

然后，直接使用 Django 提供的脚手架工具 django-admin 来创建一个项目：

```
$ django-admin startproject watering_web
```

复制代码

这时，命令会创建一个名称为 watering_web 的目录。目录中包含 manage.py、settings.py、urls.py、wsgi.py 和 asgi.py 等几个项目“骨架”文件，其中：

manage.py 是项目管理脚本，比如创建子应用、运行开发服务器等都可以通过它实现。

settings.py 是项目的整体配置文件。比如子应用的配置、数据库的配置等。

urls.py 是项目的全局路由声明文件。

wsgi.py 是 WSGI（Python Web Server Gateway Interface 的缩写）服务接口文件，是整个 Django 应用的调用入口。

asgi.py 是 ASGI（Asynchronous Server Gateway Interface 的缩写）服务接口文件。ASGI 是 WSGI 的替代者，它增加了异步应用的能力。除了支持 WSGI 协议，同时对 Websocket 和 HTTP2.0 这些长连接方式的协议提供了支持。

现在，我们进入 watering_web 目录，运行下面的命令，就可以在电脑上启动这个项目的开发服务器了。

```
$ python manage.py runserver
```

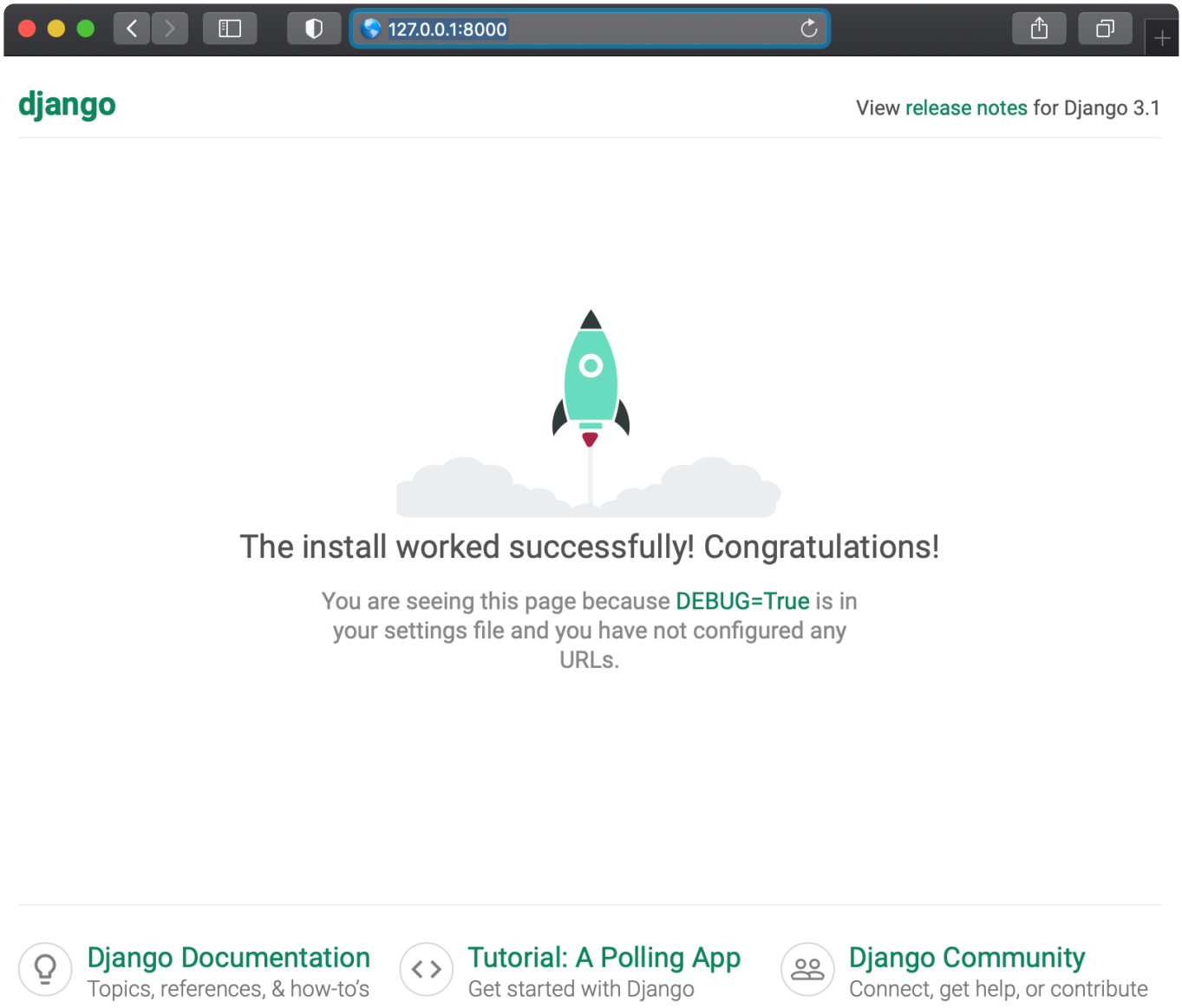
复制代码

```
➜  /Users/rubin/study/iot/geektime/watering_web  >python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply
the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
December 28, 2020 - 10:27:10
Django version 3.1.4, using settings 'watering_web.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

这非常有利于你的开发调试工作。比如，你可以在浏览器输入 http://127.0.0.1:8000/ 随时访问 Web 应用的开发效果。



当你在浏览器上看到这个界面时，就说明 Django 应用的开发环境已经准备好了。下面，我们就来开发 Web 应用。

Django 应用开发

创建子应用 watering

首先，我们通过运行下面的命令，创建 watering_web 项目的子应用 watering：

```
$ python manage.py startapp watering
```

复制代码

这时，你应该可以看到项目目录下，新增加了 watering 的目录。目录中包含了 watering 子应用的基本代码模块。

admin.py 是后台管理应用的配置文件，我们可以在其中增加数据库模型对象，让 Django 应用管理员能通过后台管理页面进行编辑。你在浏览器输入 http://127.0.0.1:8000/admin/可以访问到后台管理应用。

apps.py 是子应用的配置文件。

migrations 是数据库迁移的文件目录，它下面会保存每次数据库迁移的中间文件。

models.py 是定义子应用的数据库数据模型文件。

tests.py 是子应用的单元测试代码文件。

views.py 是子应用的视图代码文件。

理解 Django 的视图调用

在 Django 里面，网页或者说 HTTP 的响应都是视图生成的，每个视图对应 views.py 中的一个函数。Django 会根据 HTTP 请求的 URL 域名地址来选择相应的视图，而 URL 和视图函数的映射关系是在 urls.py 文件中定义的。

比如，你可以打开 watering 目录下的 views.py 文件，添加下面的函数：

复制代码

```
from django.http import HttpResponse

def demo(request):
    return HttpResponse('Hello World!')
```

然后，在 watering 目录下，新建一个 urls.py 文件，文件内容如下：

复制代码

```
from django.urls import path

from . import views

urlpatterns = [
    path('demo', views.demo, name='demo'),
]
```

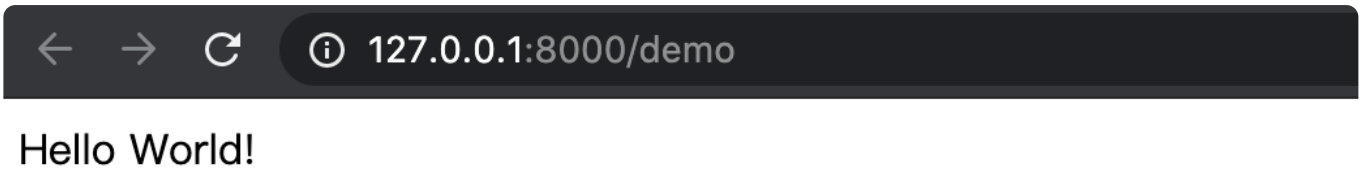
最后，在 watering_web 目录下的项目全局路由文件中，将 urlpatterns 的内容替换为下面的内容：

复制代码

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('watering.urls')),
]
```

现在，你在浏览器输入地址 http://127.0.0.1:8000/demo，就可以看看我们刚刚定义的 demo 视图的内容。



应用代码开发

接下来，我们正式开始子应用的开发。

首先，我们需要定义数据库的数据模型。Django 框架实现了 ORM（Object-Relational Mapping，对象关系映射器）技术。你可以直接在 Python 代码中定义数据库的表结构。

基于这个定义，Django 的 migration 工具能自动在数据库中创建相应的数据库表。在后面的部署阶段，我会讲解到具体的 migration 命令。

我把 models.py 的代码贴在文稿中，供你参考：

复制代码

```
from django.db import models

# Create your models here.
class Watering(models.Model):
    seq_no = models.IntegerField(blank=False, null=False)
```

```
    device_name = models.CharField(max_length=64, blank=False)
    product_id = models.CharField(max_length=64, blank=False)
    power_switch = models.IntegerField(default=0)
    humidity = models.IntegerField(default=0)
    env_temp = models.FloatField(default=0.0)
    env_hum = models.IntegerField(default=0)
    env_illum = models.IntegerField(default=0)
    timestamp = models.DateTimeField(auto_now=False)

    def __str__(self):
        return self.device_name + ' ' + str(self.seq_no)
```

有了数据模型之后，你就可以在 views.py 文件中开发视图，接收腾讯云物联网平台推送的 HTTP 请求，并且将数据存储到数据库中。

文稿中是我的示例代码，供你参考：

复制代码

```
from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse, HttpResponseBadRequest
from django.views.decorators.csrf import csrf_exempt
from django.forms.models import model_to_dict
import json
from datetime import datetime
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentC
from tencentcloud.iotexplorer.v20190423 import iotexplorer_client, models

SECRET_ID = "你的Secret ID"
SECRET_KEY = "你的Secret Key"
PRODUCT_ID = "你的ProductID"

from .models import Watering

def build_response(resp, msg):
    dict = {
        "resp": resp,
        "msg": msg,
    }
    return HttpResponse(json.dumps(dict), content_type='application/json')

def demo(request):
    return HttpResponse('Hello World!')

```

```
@csrf_exempt
def data_sync(request):
    if request.method == 'POST':
        json_data = json.loads(request.body.decode())
    else:
        return HttpResponseBadRequest('Bad Request')

    if 'seq' in json_data:
        _seq = json_data['seq']
    if 'devicename' in json_data:
        _device_name = json_data['devicename']
    if 'productid' in json_data:
        _product_id = json_data['productid']
    if 'timestamp' in json_data:
        _timestamp = json_data['timestamp']
    if 'payload' in json_data:
        _payload = json_data['payload']
        if 'method' in _payload and 'report' == _payload['method']:
            _params = _payload['params']
            _env_temp = _params['env_temp']
            _env_hum = _params['env_hum']
            _env_illum = _params['env_illum']
            _humidity = _params['humidity']
            _power_status = _params['power_switch']

            try:
                Watering.objects.create(
                    seq_no=_seq, device_name=_device_name, product_id=_product
                    power_switch=_power_status, humidity=_humidity,
                    env_temp = _env_temp, env_hum = _env_hum, env_illum = _env
                    timestamp=datetime.fromtimestamp(_timestamp) )

            except:
                return HttpResponse('Insert Failed!')

    return HttpResponse('OK')

@csrf_exempt
def latest_data(request):
    if request.method == 'POST':
        json_data = json.loads(request.body.decode())
    else:
        return HttpResponseBadRequest('Bad Request')

    if 'devicename' in json_data:
        _device_name = json_data['devicename']
    else:
        _device_name = None

    if _device_name is not None:
        try:
            data = Watering.objects.filter(device_name=_device_name).latest('t
```

```
            data_dict = model_to_dict(data, fields=['seq_no','device_name','pr
            data_dict['timestamp'] = str(data.timestamp)
            dict = {
                "resp": 0,
                "msg": "OK",
                "data": data_dict
            }
            return HttpResponse(json.dumps(dict), content_type='application/js
        except Exception as e:
            return build_response(1, str(e))
    else:
        return build_response(1, 'Parameter invalid.')
```

为了能够控制自动浇花器的水泵打开和关闭，你也可以增加发送控制命令的视图。同样，你可以参考下面的代码：

复制代码

```
@csrf_exempt
def control_device(request):
    if request.method == 'POST':
        json_data = json.loads(request.body.decode())
    else:
        return HttpResponseBadRequest('Bad Request')

    if 'devicename' in json_data:
        _device_name = json_data['devicename']
    else:
        _device_name = None
    if 'status' in json_data:
        _status = json_data['status']
    else:
        _status = None

    if _device_name is not None and _status is not None:
        try:
            cred = credential.Credential(SECRET_ID, SECRET_KEY)
            httpProfile = HttpProfile()
            httpProfile.endpoint = "iotexplorer.tencentcloudapi.com"

            clientProfile = ClientProfile()
            clientProfile.httpProfile = httpProfile
            client = iotexplorer_client.IotexplorerClient(cred, "ap-guangzhou"

            req = models.ControlDeviceDataRequest()
            data = {
                "power_switch": _status
            }
            data_str = json.dumps(data)

```

```
            params = {
                "DeviceName": _device_name,
                "ProductId": PRODUCT_ID,
                "Data": data_str
            }
            req.from_json_string(json.dumps(params))

            resp = client.ControlDeviceData(req)

            return build_response(0, resp.to_json_string())

        except TencentCloudSDKException as err:
            return build_response(1, str(err))

    else:
        return build_response(1, 'Parameter invalid.')
```

完成视图的开发后，你需要更新 urls.py 文件中的 urlpatterns，为新增视图添加映射关系。

复制代码

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('geektime/', include('watering.urls')),
]
```

同时，在 watering 子应用的目录增加一个 urls.py 文件，内容如下：

复制代码

```
from django.urls import path

from . import views

urlpatterns = [
    path('demo', views.demo, name='demo'),
    path('data', views.data_sync, name='data_sync'),
    path('control', views.control_device, name='control_device'),
    path('fetch', views.latest_data, name='latest_data'),
]
```

然后，我们来开发一个简单的网页，实现设备信息的显示和控制。为了便于理解，我这里只使用最基础的方法。当熟悉基本原理之后，你可以尝试使用 VUE、React 等前端应用框架来开发功能更丰富的 Web 前端应用。

我把代码贴在文稿中，供你参考：

复制代码

```
<!doctype html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta content="webkit" name="renderer">
<meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
<meta content="IE=Edge,chrome=1" http-equiv="X-UA-Compatible">
<style type="text/css">
    th {
    text-align: left;
    }
</style>

<title>Plant Watering</title>
<script src="jquery-3.1.1.min.js"></script>
</head>

<body>
    <h1>Plant Watering</h1>

    <table>
        <tr>
            <th>Device Name: </th><th id='devicename'></th>
        </tr>

        <tr>
            <th>Date: </th><th id='datetime'></th>
        </tr>

        <tr>
            <th>Soil Moisture (&#37;): </th><th id='moisture'></th>
        </tr>

        <tr>
            <th>Environment Temperature (℃): </th><th id='envtemp'></th>
        </tr>

        <tr>
            <th>Environment Humidity (&#37;): </th><th id='envhum'></th>
        </tr>

```

```
        <tr>
            <th>Environment Illumination (Lux): </th><th id='envillum'></th>
        </tr>

    </table>

    <button id='on'>Open</button>
    <button id='off'>Close</button>

    <script type='text/javascript'>
        var timer = false
        var interval = 5*1000   //5 seconds
        var device_name = 'Watering_1'

        $("#on").on('click',function(){
            control(1)
        })
        $("#off").on('click',function(){
            control(0)
        })

        function control(state){

            var payload = {
                "devicename":device_name,
                "status":state
            }

            $.ajax({
                url : "http://159.75.214.14/geektime/control",
                contentType: "application/json; charset=utf-8",
                method : 'POST',
                dataType: "json",
                data: JSON.stringify(payload),

                success:function (obj) {

                    if(obj.resp === 0){
                        console.log("pump on");
                    }
                    else{
                        console.log(obj.msg);
                    }
                },
                error:function(e){
                    console.log("control device post failed.");
                }
            })
        }

        function update_data(){
            var payload = {
```

```
                "devicename":device_name,
            }

            $.ajax({
                url : "http://159.75.214.14/geektime/fetch",
                contentType: "application/json; charset=utf-8",
                method : 'POST',
                dataType: "json",
                data: JSON.stringify(payload),

                success:function (obj) {

                    if(obj.resp === 0){
                        console.log("fetch success");
                        $("#devicename").text(device_name)
                        $("#datetime").text(obj.data.timestamp)
                        $("#moisture").text(obj.data.humidity)
                        $("#envtemp").text(obj.data.env_temp)
                        $("#envhum").text(obj.data.env_hum)
                        $("#envillum").text(obj.data.env_illum)
                    }
                    else{
                        console.log(obj.msg);
                    }
                },
                error:function(e){
                    console.log("fetch latest data failed.");
                }
            })
        }

        window.onload = function(){
            update_data()   //Fetch data to get devicename first.
            timer = setInterval(function(){
                update_data()
            }, interval)
        }
    </script>
</body>
</html>
```

对于这些前端代码文件，你可以在 watering_web 项目的根目录下创建一个 web 目录，然后把它们放入这个目录下。

关于 Django 项目的 settings.py 文件，你需要在 ALLOWED_HOSTS 中增加自己服务器的 IP 地址。

```
ALLOWED_HOSTS = [
    '159.75.214.14',    #替换自己的云服务器IP
]
```

复制代码

同时，在数据库的配置中，修改为 MySQL 的配置内容：

复制代码

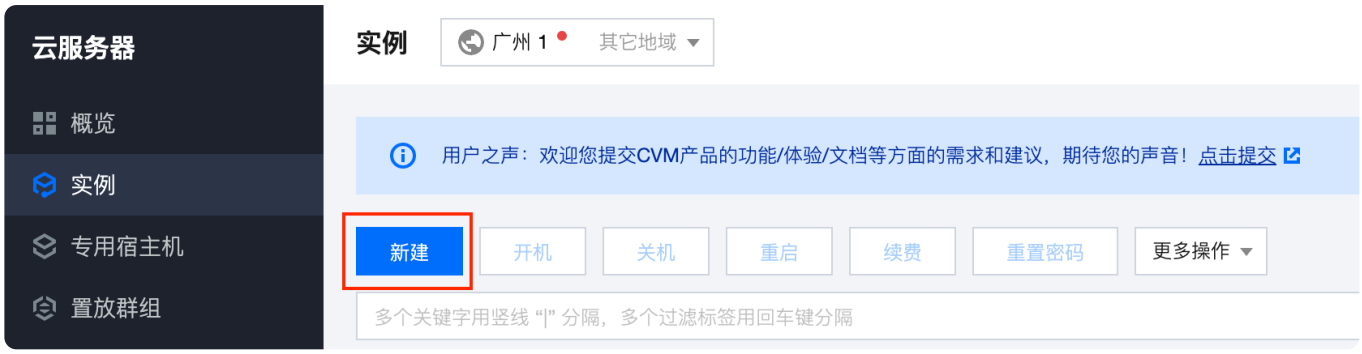
```
DATABASES = {
    # 'default': {
    #     'ENGINE': 'django.db.backends.sqlite3',
    #     'NAME': BASE_DIR / 'db.sqlite3',
    # }
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'geektime',
        'USER': 'root',
        'PASSWORD': 'geektime',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}
```

Django 应用部署

完成 Django 应用的开发，我们就可以把它部署到云服务器中。

首先，你需要先登录到腾讯云控制台，从“云产品”中点击选择“云服务器”，进入云服务的配置页面。

在云服务器页面，点击“新建”，购买一台服务器。



在新建的页面，你可以根据自己的需求选择服务器的硬件配置。如果只是用来练习，选择最低的配置就行了。

服务器的操作系统选择“Ubuntu Server 16.04.1”，其他选项保持默认值。

点击“立即购买”，完成支付后，我们重新进入云服务器控制台，就可以看到我们的服务器新实例了。

准备好云服务器后，我再介绍一下 Django 应用在 Ubuntu 服务器上的具体部署操作。

以下的介绍都是针对单台服务器部署展开的。如果多台机子部署，你需要做相应的调整，比如使用腾讯云的 CDB（云数据库服务），你就需要对 Django 应用的设置文件（settings.py）中数据库部分作修改。

Ubuntu 准备

首先，通过 SSH 登录到 Ubuntu 服务器，像连接树莓派一样，你仍然可以使用 Putty 或者 SecureCRT 这样的终端软件。

服务器的 IP 地址可以从腾讯云的云服务器控制台获取。关于用户名和密码，“腾讯云助手”公众号会在购买云服务器时发送消息通知。如果没有收到，你可以在控制台重置密码。

登录后，首先需要更新 apt，你可以运行下面的命令：

```
1 $ sudo apt-get update
```

复制代码

代码上传

接着，你需要将 Django 应用的代码上传到服务器某个目录下，比如 /home/ubuntu/iot/ 这个目录下。这涉及到下面介绍的 Nginx 和 uWSGI 的配置文件中的路径，因此，**如果代码的路径不是这个，你需要相应地修改这些配置中的路径。**

上传的工具，你还是可以使用第 19 讲中提到的 FileZilla 等软件。

数据库

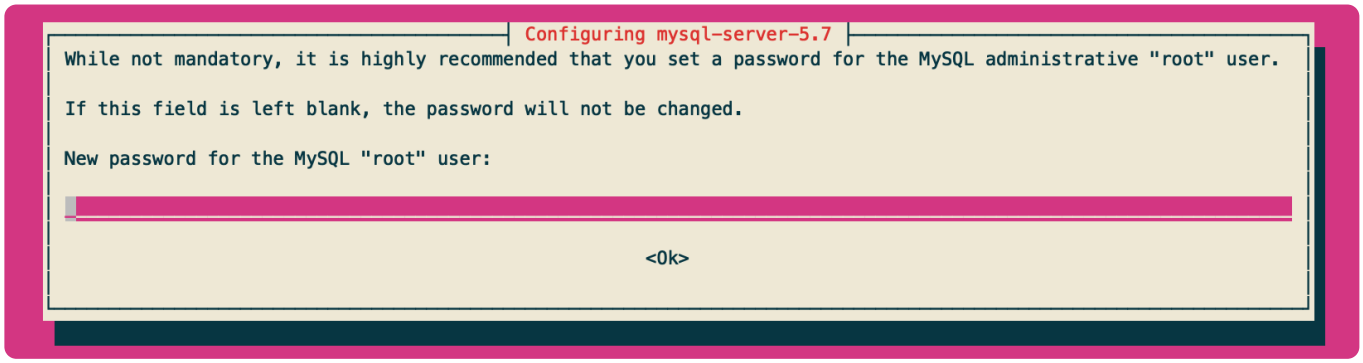
我们使用的 MySQL 数据库，可以通过以下命令来安装：

```
1 $ sudo apt-get install mysql-server mysql-client libmysqlclient-dev
```

复制代码

说明一下，如果没有安装 libmysqlclient-dev 的话，接下来安装 mysql-python 的步骤可能会报错。

安装过程中，终端也提示你输入 MySQL 数据库的密码，你可以像我一样，输入“geektime”。



安装完成后，你可以通过下面的命令，连接上数据库。

```
$ mysql -p -u root
```

然后，在数据库的交互命令行中，你需要输入下面的命令，创建 geektime 数据库。

```
create database geektime;
exit;
```

Python 环境

接下来，我们需要配置 Python 语言环境，因为应用程序和 Django 应用框架都是基于 Python 的。

首先，我们修改一下系统的默认 python，将它修改为 python3 版本。运行下面的命令：

```
$ vim ~/.bashrc
```

在文件中，增加下面的内容：

```
alias python='/usr/bin/python3'
```

复制代码

添加完成后，退出 vim。在终端输入下面的命令，使其生效：

```
$ source ~/.bashrc
```

复制代码

接着，我们需要安装 pip， Python 包管理器（Python package manager）。你可以运行下面的命令：

```
$ sudo apt-get install python3-pip
$ sudo pip3 install --upgrade pip
```

复制代码

接着，你还需要安装 python-dev， 一些软件包的安装需要依赖它。命令如下：

```
$ sudo apt-get install python3-dev
```

复制代码

后面，你需要进入 Django 应用目录（我们这里是 /home/ubuntu/iot/watering_web），然后运行下面的命令，安装所有依赖的软件包。：

```
pip3 install -r requirements.txt
```

复制代码

如果出现 locale.Error: unsupported locale setting 的错误，请在命令行输入下面的命令：

```
export LC_ALL="en_US.UTF-8"
```

复制代码

requirements.txt 文件的内容如下：

```
tencentcloud-sdk-python==3.0.313
mysqlclient
Django
```

现在，环境已经准备就绪，下面我来讲解一下数据库和 uWSGI、Nginx 的配置。

数据库 Migrate

在 Django 中，数据库的创建已经处理得非常简单，框架本身做了很多的工作。

首先，在 manage.py 文件所在的目录，运行下面的命令：

```
$ python manage.py makemigrations
```

接着，只需要运行下面的命令，就可以完成所有的数据库表创建工作。

```
$ python manage.py migrate
```

确认测试

现在，你可以执行下面的命令，启动 Django 应用。然后，你可以通过浏览器确认 Django 应用是否可以正常运行。

```
$ python manage.py runserver 0.0.0.0:8080
```

下面，我们就可以安装 uWSGI 和 Nginx 了。

uWSGI 安装

uWSGI 相当于是 Django 应用和 Nginx 之间的桥梁，它使用标准的 WSGI 接口与应用通信。你需要运行命令，安装 uWSGI：

复制代码

```
$ sudo apt-get install -y uwsgi
$ sudo apt-get install uwsgi-plugin-python3
$ pip3 install uwsgi
```

然后，在 /home/ubuntu/iot/config 目录下为 uWSGI 增加配置文件，文件内容如下：

复制代码

```
[uwsgi]
socket = 127.0.0.1:3031
chdir = /home/ubuntu/iot/watering_web
wsgi-file = watering_web/wsgi.py
processes = 4
plugins = python3
threads = 2
stats = 127.0.0.1:9191
```

接着，运行下面的命令，查看 uWSGI 是否可以正常运行。

复制代码

```
$ sudo uwsgi --ini ./config/uwsgi.ini
```

之后，我们还需要配置基于 systemd 的 uWSGI 的自启动流程。这需要创建 systemd 的 unit 文件。我们还是在 /home/ubuntu/config 目录下增加 uwsgi.service 文件，内容如下：

复制代码

```
[Unit]
Description=uWSGI
After=syslog.target

[Service]
User=ubuntu
ExecStart=/usr/bin/uwsgi --ini /home/ubuntu/iot/config/uwsgi.ini
Restart=always
KillSignal=SIGQUIT
```

```
Type=notify
StandardError=syslog
NotifyAccess=all

[Install]
WantedBy=multi-user.target
```

这时，你还不需要添加这个 service，在后面我们会通过命令添加、执行这个配置文件。

安装 Nginx

首先，你需要执行下面的命令，安装 Nginx 软件。

复制代码

```
$ sudo apt-get install nginx
```

然后，我们需要配置它。我们通过新添加文件，来配置我们的服务，在 /etc/nginx/conf.d/ 目录下，增加 iot.conf 文件，文件的内容如下:

复制代码

```
server {
    listen 80;
    #listen [::]:80 default_server;
    server_name 159.75.214.14; #替换为自己服务器IP地址

    location / {
        root /home/ubuntu/iot/watering_web/web/;
        index index.html;
    }

    location ^~ /geektime/ {
        include uwsgi_params;
        uwsgi_pass 127.0.0.1:3031;
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

并且，将 /etc/nginx/nginx.conf 文件中的“user www-data;” 修改为“user ubuntu;”。

现在，你需要执行下面的命令检查 Nginx 配置文件的语法：

```
$ sudo nginx -t
```

复制代码

如果没有错误，就可以重启 Nginx，来加载新的配置文件：

```
$ sudo service nginx restart
```

复制代码

然后，把 uWSGI 的 service 配置文件拷贝到 systemd 配置目录下：

```
$ sudo cp uwsgi.service /etc/systemd/system/
```

复制代码

现在，你可以执行下面的命令，启动 uWSGI 服务：

```
$ sudo systemctl start uwsgi
```

复制代码

如果一切正常，我们就可以把 uWSGI 添加到开机自启动中：

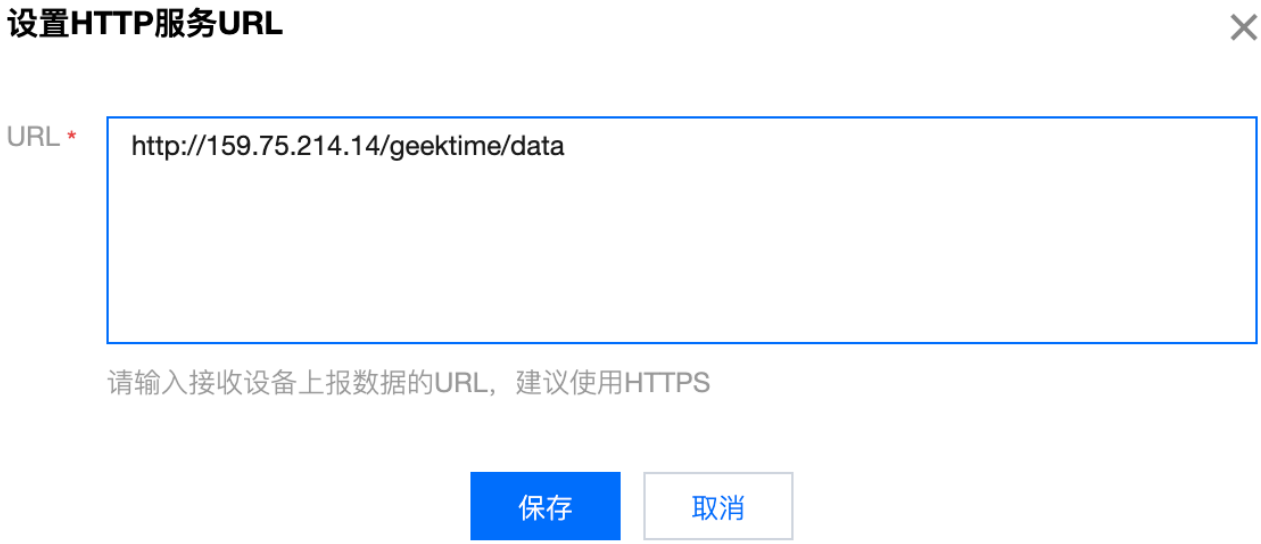
```
$ sudo systemctl enable uwsgi
```

复制代码

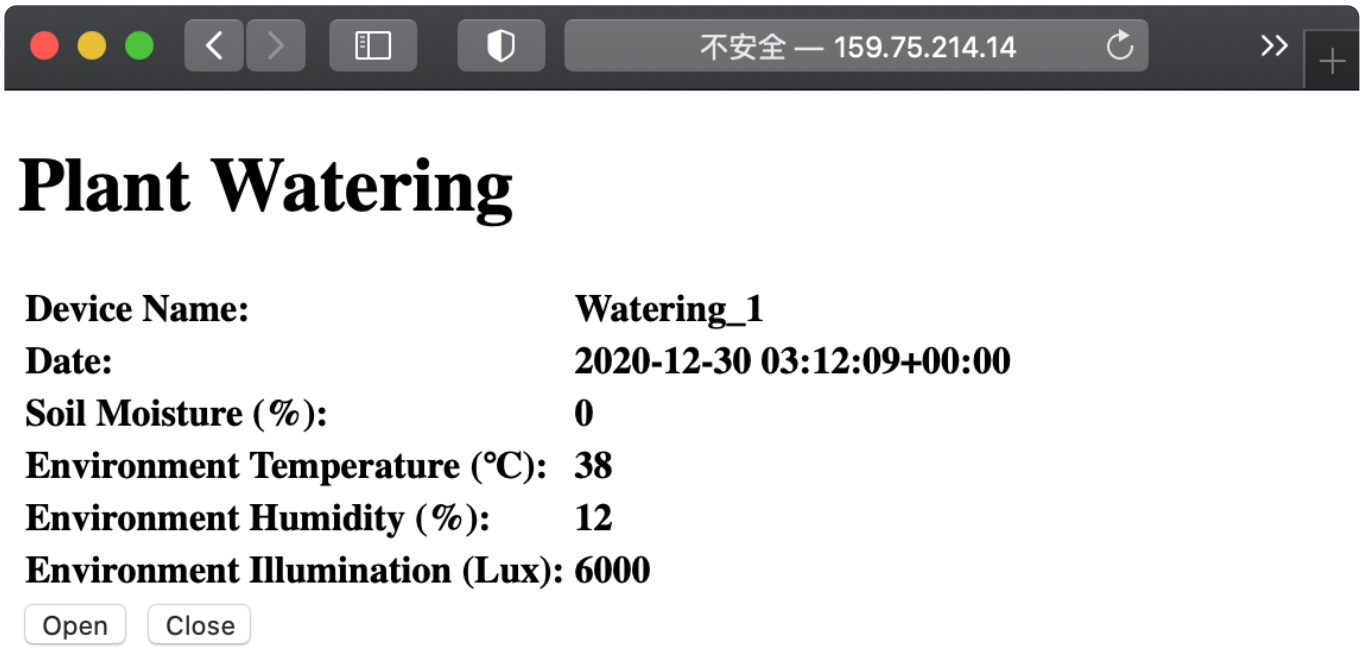
到这里，服务就部署完毕了。

云平台数据同步 URL 更新

云服务器上的 Web 应用运行正常后，你可以对腾讯云物联网平台上的 HTTP 数据同步进行更新。修改配置中的 URL 地址为自己云服务器的视图地址。



更新完成后，你就可以在浏览器上访问自己的服务器地址，查看自动浇花器的实时数据，并且控制它进行浇水。



小结

总结一下，在这一讲中，我围绕自动浇花器讲解了智能家居系统中设备数据的应用系统开发方法。主要的内容有：

1. 腾讯云物联网平台为我们提供了简单易用的数据开发方法。基于可视化界面，我们编辑“输入”，“处理”和“输出”就可以完成一个数据流的创建，实现数据的过滤和消息推送。
2. 我们可以基于物联网平台的数据同步能力，比如 HTTP 推送，实现物联网平台和应用服务器的对接。在应用服务器上，我们可以灵活地开发数据应用系统。
3. Django 是非常流行的，基于 Python 语言的 Web 应用开发框架。工作中，你可以使用 Django 比较快速地实现一个 Web 应用系统。

在这一讲中，我们的数据应用系统只包含了自动浇花器的数据，你在时间允许的情况下，也可以尝试在这个系统中增加一下智能电灯和光照传感器的数据。

另外，物联网平台的数据同步也提供了 CKafka（Cloud Kafka）的数据转发方式。通过订阅 Topic 主题消息，我们可以消费 Kafka 的消息，也可以直接将 Kafka 消息转储到 MySQL 云数据库中。

不过，Kafka 的费用比云服务器要贵不少，这里就不介绍了。如果你的条件允许，也可以动手实践一下这种方式。

这里，我整理了一个思维导图，供你参考：

思考题

最后，我给你留一个思考题吧。

在 Django 应用开发的介绍部分，我提到了 admin.py 是后台管理应用的配置文件，而且访问 http://127.0.0.1:8000/admin/ 地址，就会出现管理员登录页面。你知道在 Django 项目中如果创建管理员账号吗？另外，如何在 admin.py 文件中增加 Watering 数据库模型对象，实现对自动浇花器监测数据的查询呢？

欢迎你在留言区写一写自己的思考，同时，也欢迎你将这一讲分享给对 Web 应用开发感兴趣的朋友，大家一起交流学习。

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 21 | 多传感器集成：浇花怎么实现自动化?

下一篇 结束语 | 永远做一个具有极客精神的人

精选留言 (2)

写留言

柠檬水
2021-01-06

越来越佩服老师了，真的是全栈啊，从硬件到软件，从前端到后端。

Dan
2021-01-04

你知道在 Django 项目中如he创建管理员账号吗?
python manage.py cratesuperuser

如何在 admin.py 文件中增加 Watering 数据库模型对象，实现对自动浇花器监测数据的查询呢？...
展开

作者回复: Great!